

PREDICTING AMINO ACID INTERACTION NETWORK USING ARTIFICIAL BEE COLONY



Inspiring Excellence

Supervisor: Dilruba Showkat

Conducted By:

Mirja Shahriar Enan - 11101037

Abdullah Al Noman - 11101046

Declaration

We, hereby declare that this thesis is based on the results found by ourselves. Materials of work found by other researcher are mentioned by reference. This Thesis, neither in whole or in part, has been previously submitted for any degree.

Signature of Supervisor

Signature of Author

Dilruba Showkat

Mirja Shahriar Enan

Abdullah Al Noman

Acknowledgement

This work was suggested by Ms. Dilruba Showkat, SECS Dept., BRAC University, as a Graduation thesis. This is the work of Mirja Shahriar Enan and Abdullah Al Noman, students of the SECS department of BRAC University, studying CSE and CS respectively starting from the year 2011. The document has been prepared as an effort to compile the knowledge obtained by us during these four years of education and produce a final thesis which innovatively addresses one of the issues of research in bioinformatics field, protein folding.

We would like to express our gratitude to Almighty Allah (SWT) who gave us the opportunity, determination, strength and intelligence to complete our thesis.

We would like to thank our supervisor, Ms.Dilruba Showkat sincerely for her consistent supervision, guidance and unflinching encouragement in accomplishing our work.

We also want to acknowledge Mr. Shiplu Hawlader, Lecturer at Department of Computer Science & Engineering, University of Dhaka, who has consistently shown his interest in our work and assisted us in its completion.

Abstract

Proteins are biological macromolecules participating in the large majority of processes which govern organisms. Amino Acids are the building block of protein and its primary structure is a linear chain of amino acids, which lacks functionality. Amino acids interact with each other to produce a three-dimensional structure, known as protein folding. In our proposed approach, we are considering protein as a graph of amino acid interaction network whose vertices are the amino acids and edges are the interaction between them. We proposed a single objective evolutionary algorithm and a recently introduced swarm-based optimization algorithm, named Artificial Bee Colony algorithm to predict the interaction between amino acids and thus solving the protein structure prediction (PSP) problem.

TABLE OF CONTENTS

PART 1

INTRODUCTION	1
1.1 MOTIVATION	3
1.2 THESIS OUTLINE	4

PART 2

BACKGROUND RESEARCH

2.1 THEORY	
2.1.1 PROTEIN STRUCTURE	5
2.1.2 AMINO ACID INTERACTION NETWORK	8
2.1.3 THE PROTEIN FOLDING PROBLEM	11
2.2 ALGORITHMS	
2.2.1 MULTI OBJECTIVE GENETIC ALGORITHM	12
2.2.2 ANT COLONY OPTIMIZATION	14
2.2.2.1 MULTI OBJECTIVE ANT COLONY OPTIMIZATION	16
2.2.3 PARTICLE SWARM OPTIMIZATION	17
2.2.4 BAT ALGORITHM	21

PART 3

PROPOSED WORK

3.1 INTERACTION NETWORK PREDICTION	24
3.2 DATASET	25
3.3 GENETIC ALGORITHM	
3.3.1 OVERVIEW OF GA	27
3.3.2 GENETIC VARIATION OPERATORS	
3.3.2.1 SELECTION	28
3.3.2.2 CROSSOVER	28
3.3.2.3 MUTATION	29
3.3.3 PSEUDOCODE	30

3.4 CLUSTERING	32
3.5 ARTIFICIAL BEE COLONY ALGORITHM	
3.5.1 OVERVIEW OF ABC	34
3.5.2 HOW ABC WORKS	37
3.5.3 A PROPOSED CROSSOVER OPERATOR	40
3.5.4 PSEUDOCODE	41
 PART 4	
<hr/>	
PERFORMANCE ANALYSIS	
3.1 ANALYSIS OF GA	43
3.2 ANALYSIS OF ABC	43
3.3 ALGORITHM COMPLEXITY	44
3.4 RESULT	45
 PART 5	
<hr/>	
FUTURE WORK	49
 PART 6	
<hr/>	
CONCLUSION	50
 PART 7	
<hr/>	
REFERENCES	51

1. Introduction

Proteins are nature's most vital particles and are available in every single living being. The human body comprises basically of proteins of distinctive structures, motion and capacities. Oxygen is transported to the human cells by hemoglobin, a multi space protein in complex with iron which organizes the oxygen particle. The skin, the immune system, and muscle tissues are different samples of protein courses of action vital to human life. The building pieces of proteins are amino acids, little particles primarily made out of carbon, nitrogen, oxygen and hydrogen. There are 20 distinctive amino acids with particular properties that can be a part of the protein structures. The sequence of the amino acids is unique for each protein and determines its fold and the function.

Proteins are the biological macromolecules which participates in most of the process that oversees living beings. The roles played by proteins are varied and complex. The best known objective of proteins in a cell is performed as enzymes which act as catalysts and increase several orders of magnitude, with a noteworthy specificity. Like all catalysts, enzymes take part in reaction and the speed of multiple chemical reactions essential to the organism survival like DNA replication, DNA repair and transcription. Proteins are storage of a cell and transports small molecules or ions, control the passages of molecule through the cell membranes, and so forth. Hormones are another sort of proteins which transmits data and permit the regulation of complex cell forms.

Genome sequencing tasks produce an always expanding number of protein sequences. For example, the Human Genome Project has distinguished over 30,000 genes [1] which may encode about 100,000 proteins. One of the first tasks when annotating a new genome, is

to allocate functions to the proteins created by the genes. To completely comprehend the biological functions of proteins, the knowledge of their structure is crucial.

Proteins are amino acid chain that are bonded together in peptide bonds. In their natural environment, proteins adopt a local minimal three-dimensional form. This process of forming three-dimensional structure of a protein is called protein folding and this is not fully understood yet in system biology. The procedure is an aftereffect of connection between protein's amino acids which form chemical bond.

Bee colony optimization algorithm is one of the problem solving algorithms that have been applied on search optimization problems. It has been introduced by Karaboga in 2005, which is inspired by the behavior of honey bees [2]. In ABC system, artificial bees fly around in a multidimensional search space and this system combines local and global search methods attempting to balance exploration and exploitation process.

In this paper, we proposed an approach to predict the interaction network of amino acids using a combination of Genetic Algorithm (GA) and Artificial Bee Colony algorithm (ABC).

1.1 Motivation:

To be able to understand how a protein accomplishes its biological function, and to be able to act on the cellular processes in which the protein intervenes, it is essential to know its structure. Currently, protein structures are primarily determined by techniques such as MRI (magnetic resonance imaging) and X-ray crystallography, which is expensive in terms of equipment, computation and time. Additionally, these techniques require isolation, purification and crystallization of the target protein. The number of protein sequences known is much more important than the number of solved structures, so this gap continues to grow quickly. In the past, the Science Magazine named the protein folding problem (PFP) as one of the 125 biggest unsolved problems in science. The PFP addresses the question of how the amino acid sequence of a specific protein dictates its structure. We figured out that the design of methods making it possible to predict the protein structure from its sequence and found it as a fascinating one.

1.2 Thesis Outline

Section 1 consists of the *Introduction* about the protein structure prediction (PSP) and *Motivation*, why we do our thesis on such topic.

Section 2 deals with the *Background Researches*, includes theory and different optimization algorithms that we study for our work.

Section 3 is comprised of our *Proposed Work*. It has step by step procedure of how we use GA and ABC for predicting the interaction network. This section starts with how we address interaction prediction as graph network and dataset we use. After that a brief overview of GA and different genetic variation operators are discussed. Then a clustering algorithm followed by details of our proposed ABC algorithm is discussed.

Section 4 is *performance analysis* of the two algorithms and test result.

Our planned *future work* is in Section 5.

Section 6 is *conclusion* and Section 7 is *references*.

2. Background Research

2.1 Theory

2.1.1 Protein Structure

Inside of a protein is usually composed of hydrophobic amino acids, which therefore keep away from contact with the hydrophilic environment in the cell. At the protein surface, polar and hydrophilic residues dominate, as they are attracted to the polarity of the solvent. Charged amino acid side chains like those of 'Asp', 'Glu', 'Lys' and 'Arg' are hydrophilic, whereas large non-charged residues such as the aliphatic 'Leu', 'Ile' and 'Val' as well as the aromatic 'Phe', 'Tyr' and 'Trp' are hydrophobic. The backbone of proteins, which includes the alpha carbons and the peptide bond, is hydrophilic and is thus not favored in the hydrophobic core. Naturally this problem has been tackled by forming secondary structure elements (SSE) from the backbone, thus preventing their hydrophilic NH and CO groups from exposure towards the hydrophobic surroundings.

Proteins are formed with amino acids which are linked by peptide bonds to form polypeptide chain. They have intricate and unpredictable structure and can be distinguished in some levels such as -

- The amino acid sequence of a protein's polypeptide chain is called its primary or one-dimensional structure. It can be considered as a word over the 20-letter amino acid alphabet.
- Different elements of the sequence form regular secondary (2D) structured, such as α -helices & β -strands.

- The tertiary (3D) structure is formed by packing such structural elements into one or several compact globular units called domains.
- The final protein may contain several polypeptide chains arranged in a quaternary structure.

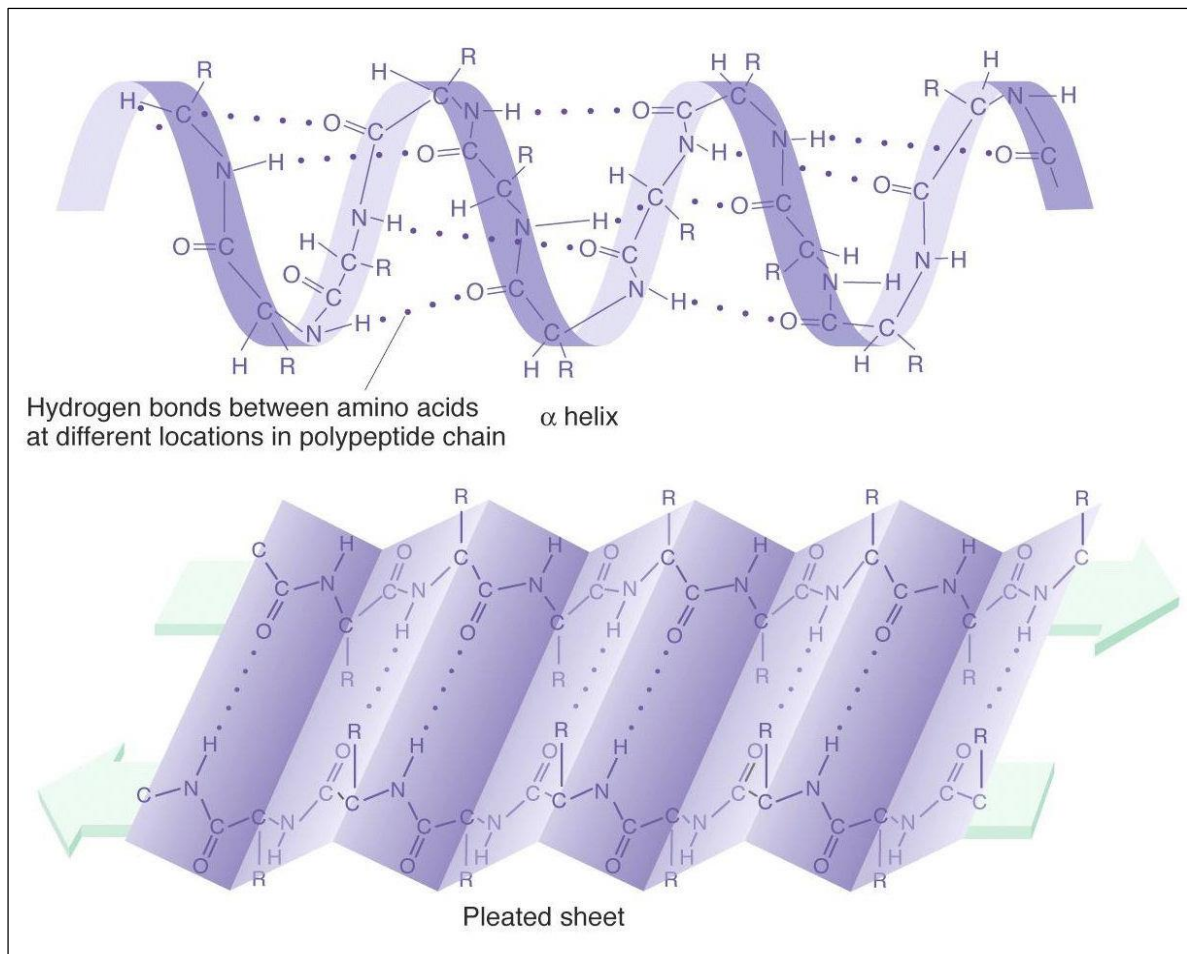


Figure 1: Hydrogen bonds between amino acids in a protein

An α helix adopts a right-handed helical conformation with 3.6 residues per turn with hydrogen bonds between $C'=O$ group of residue n and NH group of residue $n+4$.

A β -sheet is build up from a combination of several regions of the polypeptide chain where hydrogen bonds can form between C'=O groups of one β strand and another NH group parallel to the first strand. There are two kinds of β -sheet formations, anti-parallel β -sheets (in which the two strands run in opposite directions) and parallel sheets (in which the two strands run in the same direction).

Based on the local organization of the secondary structure elements (SSE), proteins are divided in the following four classes -

1. All α , proteins have only α -helix secondary structure.
2. All β , proteins have only β -strand secondary structure.
3. α / β , proteins have mixed α -helix and β -strand secondary structure.
4. $\alpha + \beta$, proteins have separated α -helix and β -strand secondary structure.

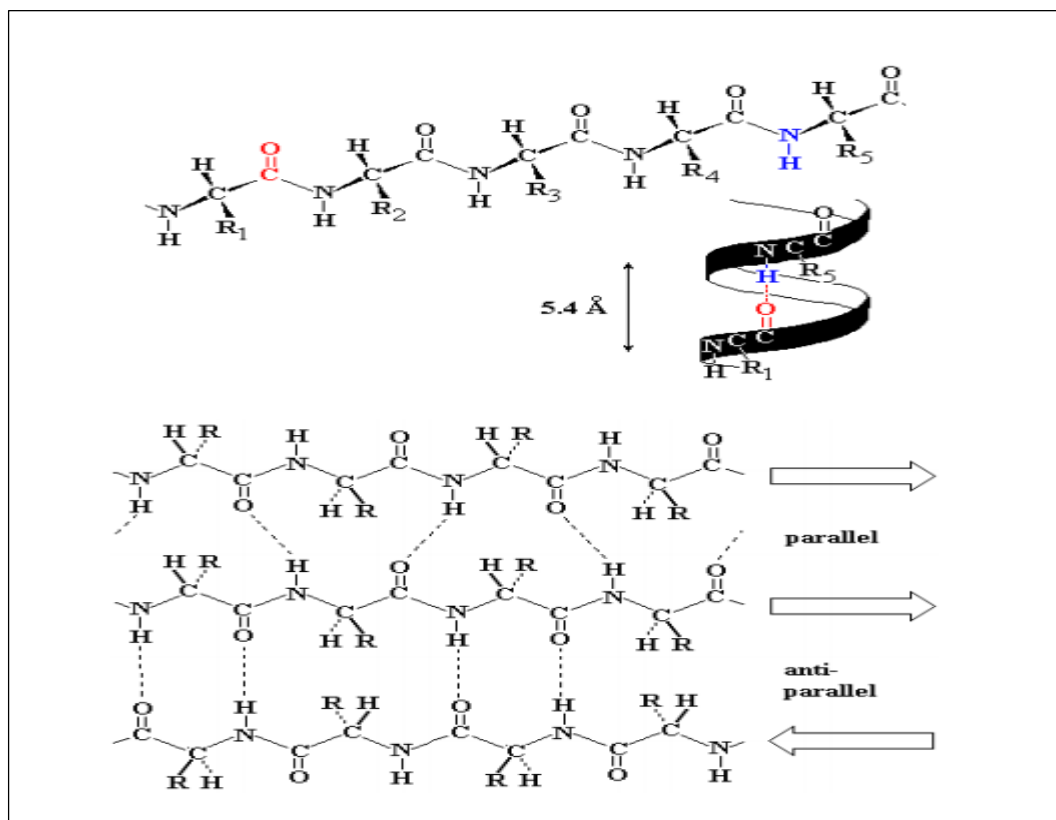


Figure 2: *Top: An α -helix illustrated as ribbon diagram,
Bottom: A β -sheet composed by three strands*

From these divisions, a more detailed classification can be done. A most frequently used one is Structural Classification of Proteins (SCOP) [3]. They are hierarchical classifications of proteins' structural domains. A domain corresponds to a part of a protein which has a hydrophobic core and not much interaction with other parts of the protein.

2.1.2 Amino Acid Interaction Network

The 3D structure of a protein is determined by the coordinates of its atoms. This information is available in Protein Data Bank (PDB) [4], which regroups all experimentally solved protein structures.

ATOM	13	CB	ARG	A	2	24.785	7.479	31.007	1.00	49.36
ATOM	14	CG	ARG	A	2	24.962	6.035	30.583	1.00	48.36
ATOM	15	CD	ARG	A	2	23.632	5.464	30.038	1.00	49.86
ATOM	16	NE	ARG	A	2	22.605	5.311	31.077	1.00	61.17
ATOM	17	CZ	ARG	A	2	21.747	6.260	31.453	1.00	60.10
ATOM	18	N	ALA	A	3	25.897	10.573	30.824	1.00	49.23
ATOM	19	CA	ALA	A	3	25.601	12.006	31.003	1.00	48.16
ATOM	20	C	ALA	A	3	24.193	12.257	31.554	1.00	47.08
ATOM	21	O	ALA	A	3	24.012	13.108	32.406	1.00	48.60
ATOM	22	CB	ALA	A	3	25.747	12.725	29.683	1.00	46.42
ATOM	23	N	ASP	A	4	23.208	11.527	31.033	1.00	47.05
ATOM	24	CA	ASP	A	4	21.825	11.701	31.460	1.00	47.05
ATOM	25	C	ASP	A	4	21.554	11.185	32.870	1.00	47.03
ATOM	26	O	ASP	A	4	20.664	11.699	33.524	1.00	47.69
ATOM	27	CB	ASP	A	4	20.831	11.107	30.464	1.00	46.90
ATOM	28	CG	ASP	A	4	20.999	9.612	30.260	1.00	50.11
ATOM	29	OD1	ASP	A	4	22.058	9.187	29.758	1.00	57.42
ATOM	30	OD2	ASP	A	4	20.040	8.869	30.549	1.00	54.19
ATOM	31	N	ALA	A	5	22.306	10.181	33.322	1.00	46.15
ATOM	32	CA	ALA	A	5	22.151	9.638	34.679	1.00	45.87
ATOM	33	C	ALA	A	5	22.759	10.627	35.662	1.00	45.34
ATOM	34	O	ALA	A	5	22.243	10.863	36.754	1.00	45.18
ATOM	35	CB	ALA	A	5	22.826	8.271	34.795	1.00	45.46
ATOM	36	N	LEU	A	6	23.856	11.237	35.245	1.00	45.52
ATOM	37	CA	LEU	A	6	24.537	12.204	36.065	1.00	46.72
ATOM	38	C	LEU	A	6	23.705	13.489	36.215	1.00	43.98
ATOM	39	O	LEU	A	6	23.698	14.088	37.263	1.00	43.09
ATOM	40	CB	LEU	A	6	25.898	12.486	35.419	1.00	48.85
ATOM	41	CG	LEU	A	6	26.777	13.577	36.011	1.00	54.48
ATOM	42	CD1	LEU	A	6	26.973	13.379	37.519	1.00	58.68
ATOM	43	CD2	LEU	A	6	28.111	13.580	35.234	1.00	52.93

Figure 3: A screenshot taken from 2O7T protein PDB file [5], red markers shows the C α atom And green markers shows their coordinate in XYZ plane.

We can compute the distance between two atoms by using the coordinates of them. We define the distance between two amino acids as the distance between their $C\alpha$ atoms. Considering the $C\alpha$ atom as a “center” of the amino acid is an approximation, but it works well enough for our purpose.

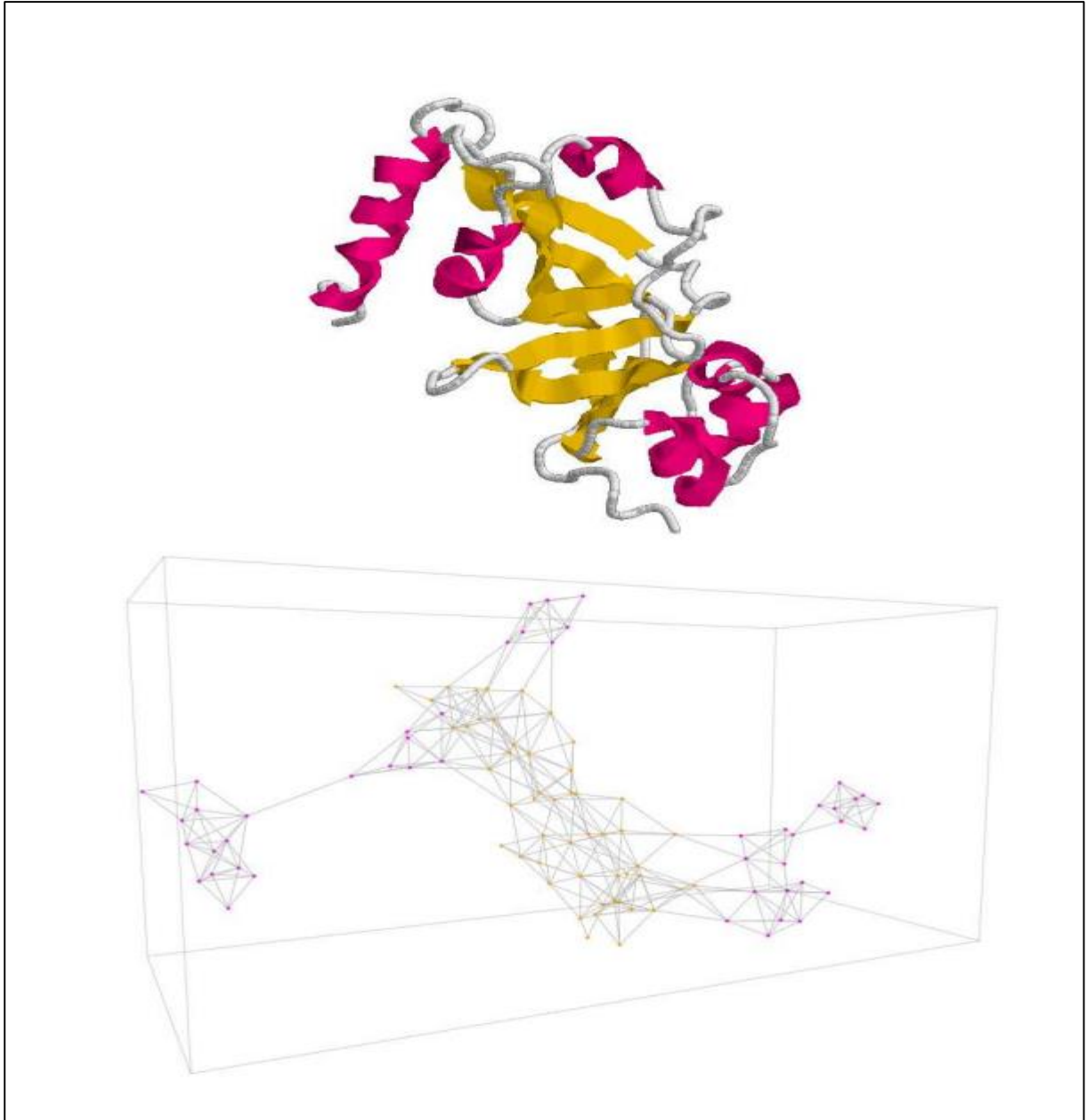


Figure 4: *Top:* Protein 1DTP,
Bottom: SSE-IN of protein 1DTP

Let us denote by N the number of amino acids in the protein. A contact map matrix will be an $N \times N$ matrix (0-1), whose element (i, j) is 1 if there is a contact between amino acids i and j and 0 otherwise. We can get useful information from this, such as - the secondary structure elements can be identified using this matrix. For example, α -helices spread along the main diagonal, while β -sheets appear as bands parallel or perpendicular to the main diagonal [6]. There are different ways to define the contact between two amino acids. One of them is that two amino acids are in contact if and only if the distance between them is below a given threshold [7]. A commonly used threshold is 7\AA and this is the value we use.

Now considering a graph with N vertices (where each vertex corresponds to an amino acid) and the contact between them, it is called contact map graph. The contact map graph is a dynamic portrayal of the protein structure considering just the collaborations between the amino acids. Then, we call the subgraph induced by the set of amino acids participating in SSE as SSE interaction network (SSE-IN) and this is the object we study in this paper. We are not considering the amino acids not participating in SSE because evolution tends to preserve the structural core of proteins composed from SSE. Then again, the loops (regions between SSE) are not so important to the structure and hence, are subject to more mutations. That is the reason homologous proteins have a tendency to have relatively preserved structural cores and variable loop regions. Thus, the structure determining interactions are those between amino acids belonging to the same SSE on local level and between different SSEs on global level. Fig. 2 gives an example of a protein and its SSE-IN. Proteins can be treated as a network of amino acids. According to the diameter value, average mean degree and clustering coefficient shown in [8], we can say a protein is a network of amino acids.

2.1.3 The Protein Folding Problem

Several tens of thousands of protein sequences are encoded in the human genome. A protein is comparable to an amino acid chain which folds to adopt its tertiary structure. Thus, this 3D structure enables a protein to achieve its biological function. Each protein must quickly find its native structure, functional, among innumerable alternative conformations. The protein 3D structure prediction is one of the most important problems of bioinformatics and remains however still irresolute in the majority of cases.

The problem is summarized by the following question: being given a protein defined by its sequence of amino acids, which is its native structure? For example - the structure whose amino acids are correctly organized in three dimensions so that proteins can achieve correctly their biological functions. As well, the native structure is considered as the most stable with a minimum energy level. Unfortunately, the exact answer is not always possible that is why the researchers have developed study models to provide a feasible solution for any unknown sequences.

However, models to fold proteins bring back to NP-Hard optimization problems [9]. Those kinds of models consider a conformational space where the modeled protein tries to reach its minimum energy level which corresponds to its native structure. Therefore, any algorithm of resolution seems improbable and ineffective, the fact is that in the absolute no study model is yet able to entirely define the general principles of the protein folding.

2.2 Algorithms

2.2.1 Multi Objective Genetic Algorithm (MO-GA)

The concept of genetic algorithms (GA) was developed in 1975 [10]. GA is inspired by the evolutionist theory explaining the origin of species. In nature, weak and unfit species within their environment are faced with extinction by natural selection. The strong ones have greater opportunity to pass their genes to future generations via reproduction. In the long run, species carrying the correct combination in their genes become dominant in their population.

Multi-objective formulations are a realistic models for many complex real world optimization problems. Customized genetic algorithms have been demonstrated to be particularly effective to determine solutions to these problems. A multi-objective decision problem is defined as follows: Given an n -dimensional decision variable vector $\mathbf{x}=\{x_1, \dots, x_n\}$ in the solution space \mathbf{X} , find a vector \mathbf{x}^* that minimizes a given set of K objective functions $\mathbf{z}(\mathbf{x}^*) = \{z_1(\mathbf{x}^*), \dots, z_K(\mathbf{x}^*)\}$. The solution space \mathbf{X} is generally restricted by a series of constraints, such as $g_j(\mathbf{x}^*)=b_j$ for $j = 1, \dots, m$, and bounds on the decision variables.

In many real-life problems, objectives under consideration conflict with each other, and optimizing a particular solution with respect to a single objective can result in unacceptable results with respect to the other objectives. For multiple-objective problems, the objectives are generally conflicting, preventing simultaneous optimization of each objective.

GA's are a popular meta-heuristic that is particularly well-suited for this class of problems. Traditional GA are customized to accommodate multi-objective problems by using specialized fitness functions. There are two general approaches, one is to combine the

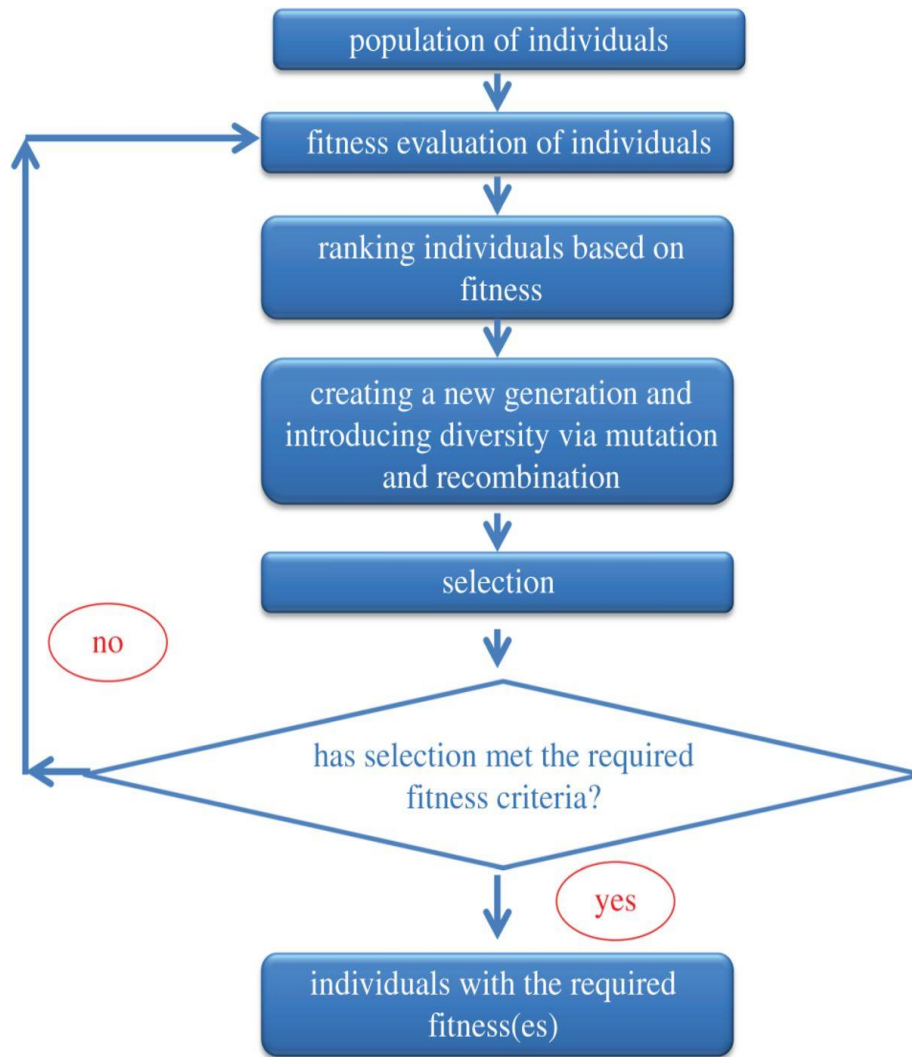


Figure 5: Flowchart of simple Genetic Algorithm

individual objective functions into a single composite function. Determination of a single objective is possible with methods such as utility theory, weighted sum method, etc., but the problem lies in the correct selection of the weights or utility functions to characterize the decision-makers preferences.

The second general approach is to determine an entire Pareto optimal solution set or a representative subset. A Pareto optimal set is a set of solutions that are non-dominated with

respect to each other. While moving from one Pareto solution to another, there is always a certain amount of sacrifice in one objective to achieve a certain amount of gain in the other.

2.2.2 Ant Colony Optimization (ACO)

The basic idea of Ant Colony Optimization (ACO) [11] is to model the problem to solve as the search for a minimum cost path in a graph, and to use artificial ants to search for good paths. The behavior of artificial ants is inspired from real ants, as they lay pheromone on components (edges and/or vertices) of the graph and they choose their path with respect to

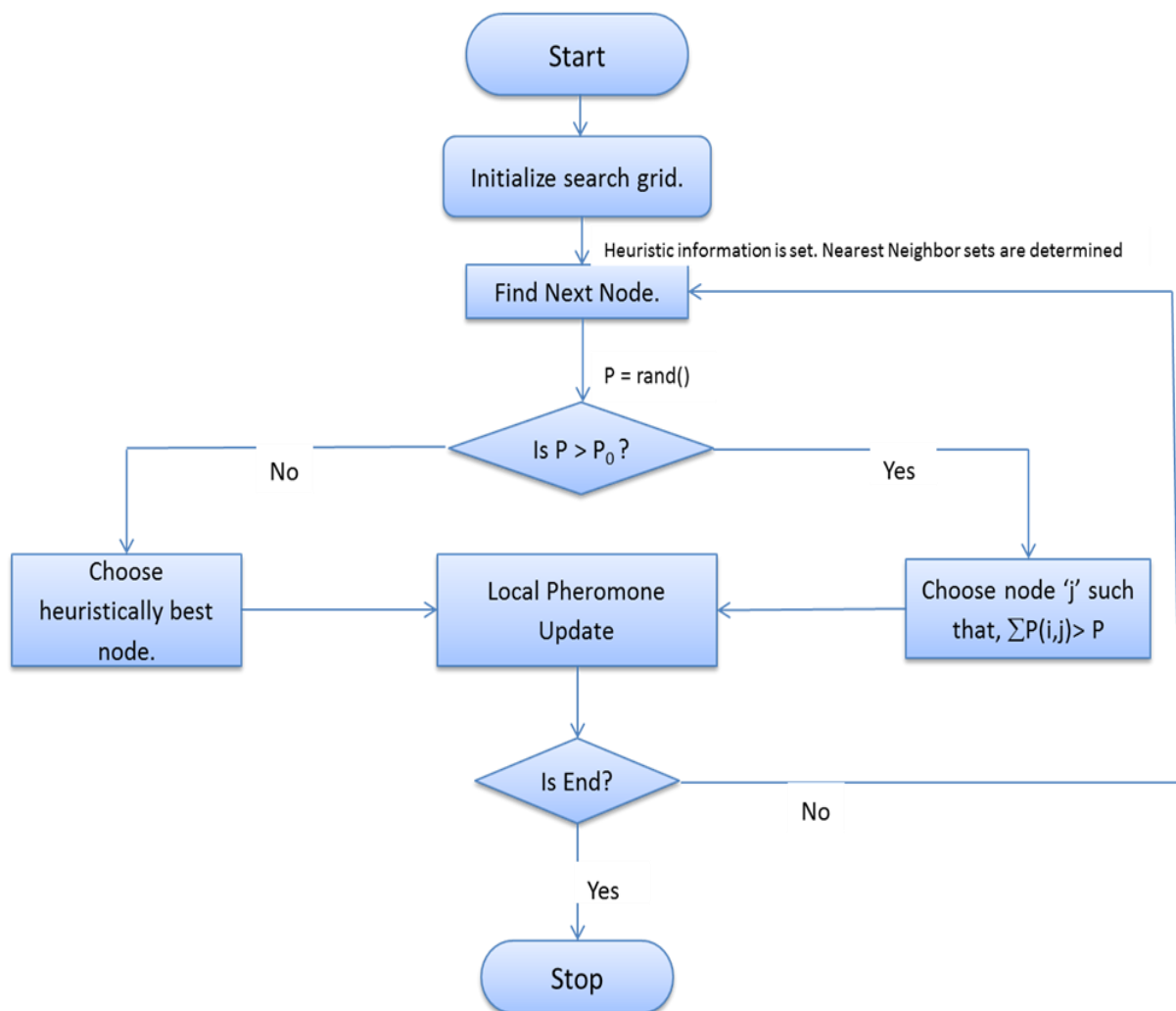


Figure 6: Ant Colony Optimization Algorithm Flowchart

probabilities that depend on pheromone trails that have been previously laid by the colony. Artificial ants also have some extra-features that do not find their counterpart in real ants. In particular, they are usually associated with data structures that contain the memory of their previous actions, and they may apply some daemon procedures, such as local search, to improve the quality of computed paths. Finally, the probability for an artificial ant to choose a component often depends not only on pheromone, but also on problem-specific local heuristics.

Initially the ants run blind until one manages to find food, upon doing so it then returns to the nest, leaving a trail of pheromones along the path it takes. Once a pheromone trail has been established, rather than running around randomly, other ants within the colony will prefer to follow the pheromone trail, leaving their own pheromones if they find food reinforcing the trail.

There are three significant characteristics with ant colony algorithms which contribute to its performance:

1. The better paths get more pheromones deposited on them and hence become more attractive to future ants. This positive feedback results in a quick divergence on good solutions [9].
2. The algorithm is distributed over a colony of ants which helps prevent the algorithm from converging too soon on a local minima [9].
3. A greedy heuristic results in good solutions being found early on, giving the algorithm a good starting point from which to build upon [9].

2.2.2.1 Multi Objective ACO (MO-ACO)

In many real-life optimization problems there are several objectives to optimize. For such multi-objective problems, there is not usually a single best solution but a set of solutions that are superior to others when considering all objectives. This set is called the Pareto set or non-dominated solutions. This multiplicity of solutions is explained by the fact that objectives are generally conflicting ones. The application of ACO principles to multi-objective optimization is a topic of research and to address the design of a multi-objective ACO algorithm (MO-ACO) for this type of problem, several concepts must be reviewed. The management of the pheromone information in MO-ACO turns out to be a complex task that involves the definition of the pheromone information, such as - which solutions are selected to update the pheromone information and how these solutions modify the pheromone information etc. In addition, some authors propose the use of multiple colonies, so that each colony weighs differently the relative importance of the multiple objectives.

Multi Objective ACO Algorithm

Set parameters

Initialize pheromone trails τ

Initialize heuristic matrix η

Initialize Pareto set \mathcal{P} as empty

While termination criteria not met **do**

 Construct Ant Solution

 Apply Local Search

 Update Pareto Set

Update Global Pheromone

Return the Pareto Set

When multiple colonies are considered, the management of the pheromone information is even more complicated. Finally, the use of local search methods must be considered. All these features can be seen as components of a certain configuration of a general MO-ACO algorithm.

2.2.3 Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) is a population-based stochastic approach for solving continuous and discrete optimization problems. It is a heuristic global optimization method put forward originally in 1995. It is developed from swarm intelligence and is based on the research of bird and fish flock movement behavior.

While searching for food, the birds are either scattered or go together before they locate the place where they can find the food. While the birds are searching for food from one place to another, there is always a bird that can smell the food very well, that is, the bird is perceptible of the place where the food can be found, having the better food resource information. Because they are transmitting the information, especially the good information at any time while searching the food from one place to another, conducted by the good information, the birds will eventually flock to the place where food can be found. As far as particle swarm optimization algorithm is concerned, solution swarm is compared to the bird swarm, the birds' moving from one place to another is equal to the development of the solution swarm, good information is equal to the most optimistic solution, and the food resource is equal to the most optimistic solution during the whole course. The most optimistic solution can be worked out in particle swarm optimization algorithm by the cooperation of each individual. The particle without quality and volume serves as each individual, and the simple behavioral pattern is

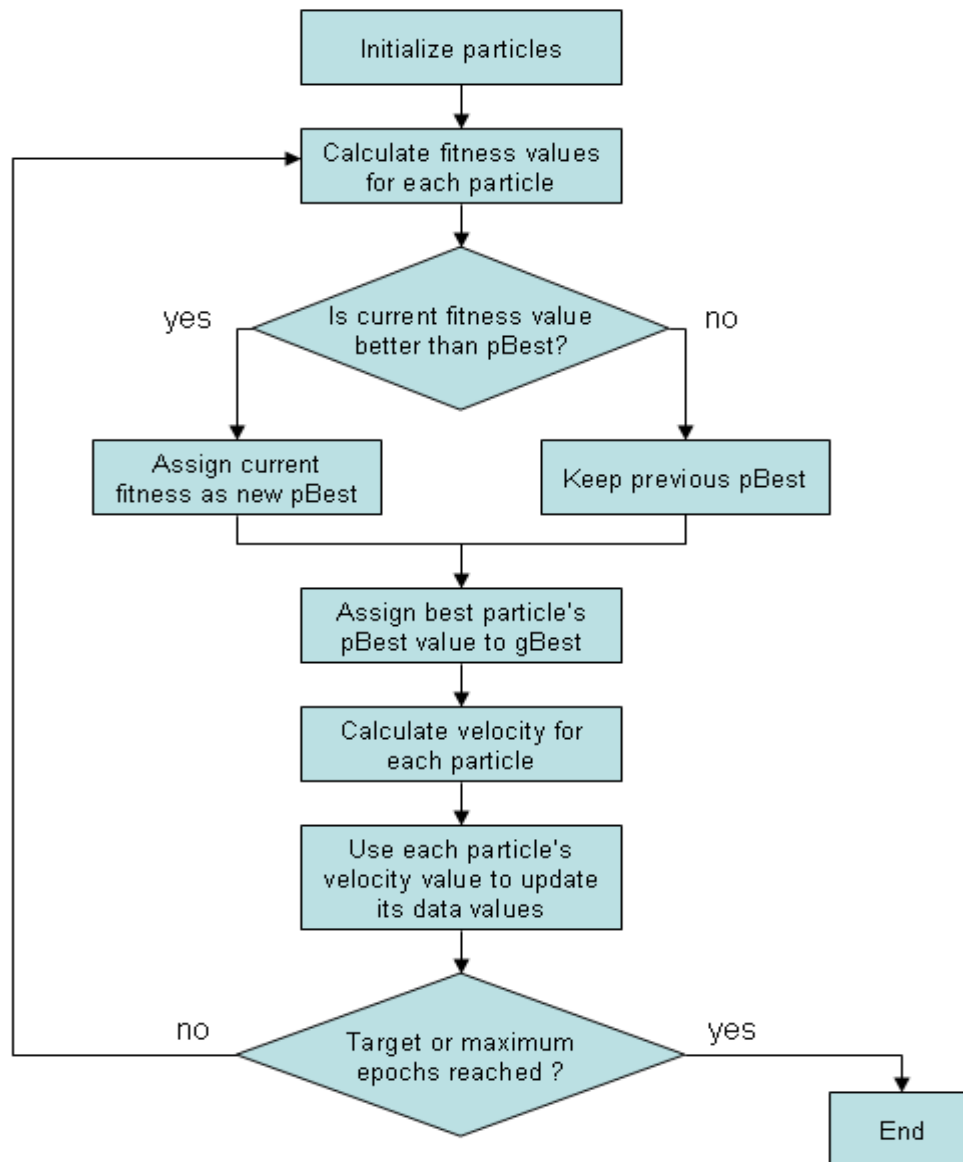


Figure 7: flowchart for basic PSO algorithm

regulated for each particle to show the complexity of the whole particle swarm. This algorithm can be used to work out the complex optimist problems.

Basic Particle Swarm Optimization Algorithm:

In the basic particle swarm optimization algorithm, particle swarm consists of “n” particles, and the position of each particle stands for the potential solution in D-dimensional space. The particles change its condition according to the following three principles:

1. to keep its inertia
2. to change the condition according to its most optimist position
3. to change the condition according to the swarm's most optimist position.

The position of each particle in the swarm is affected both by the most optimist position during its movement and the position of the most optimist particle in its surrounding (near experience). When the whole particle swarm is surrounding the particle, the most optimist position of the surrounding is equal to the one of the whole most optimist particle; this algorithm is called the whole PSO. If the narrow surrounding is used in the algorithm, this algorithm is called the partial PSO. Each particle can be shown by its current speed and position, the most optimist position of each individual and the most optimist position of the surrounding.

Particle Swarm Optimization Algorithm

```

for each particle  $i$  in  $S$  do
    for each dimension  $d$  in  $D$  do
         $x_{i,d} = Rnd(x_{min}, x_{max})$ 
         $v_{i,d} = Rnd(-v_{max}/3, v_{max}/3)$ 
    end for
     $pb_i = x_i$ 
    if  $f(pb_i) < f(gb)$ 
         $gb = pb_i$ 
    end if
end for

```

In order to avoid particle being far away from the searching space, the speed of the particle created at its each direction is confined between $-v_{max}$, and v_{max} . If the number of v_{max} is too

big, the solution is far from the best, if the number of vd_{\max} is too small, the solution will be the local optimism.

Advantages of the basic PSO:

1. PSO is based on the intelligence. It can be applied into both scientific research and engineering use.
2. PSO have no overlapping and mutation calculation. The search can be carried out by the speed of the particle.
3. The calculation in PSO is very simple. Compared with the other developing calculations, it occupies the bigger optimization ability and it can be completed easily.
4. PSO adopts the real number code, and it is decided directly by the solution. The number of the dimension is equal to the constant of the solution.

Disadvantages of the basic PSO:

1. The method easily suffers from the partial optimism, which causes the less exact at the regulation of its speed and the direction.
2. The method can not work out the problems of scattering and optimization.
3. The method can not work out the problems of non-coordinate system, such as the solution to the energy field and the moving rules of the particles in the energy field.

2.2.4 Bat Algorithm

The vast majority of heuristic and metaheuristic algorithms have been derived from the behavior of biological systems and physical systems in nature. A new metaheuristic method, namely, the Bat Algorithm (BA), based on the echolocation behavior of bats. The capability of echolocation of microbats is fascinating as these bats can find their prey and discriminate different types of insects even in complete darkness.

Behavior of Microbats:

Most microbats are insectivores. Microbats use a type of sonar, called, echolocation, to detect prey, avoid obstacles, and locate their roosting crevices in the dark. These bats emit a very loud sound pulse and listen for the echo that bounces back from the surrounding objects. Their pulses vary in properties and can be correlated with their hunting strategies, depending on the species. Most bats use short, frequency-modulated signals to sweep through about an octave, while others more often use constant-frequency signals for echolocation. Their signal bandwidth varies depends on the species, and often increased by using more harmonics.

Acoustics of Echolocation:

Each pulse only lasts a few thousandths of a second (up to about 8 to 10 ms), however, it has a constant frequency which is usually in the region of 25 kHz to 150 kHz. The typical range of frequencies for most bat species are in the region between 25kHz and 100kHz, Studies show that microbats use the time delay from the emission and detection of the echo, the time difference between their two ears, and the loudness variations of the echoes to build up three dimensional scenario of the surrounding. They can detect the distance and orientation of the target, the type of prey, and even the moving speed of the prey such as small insects. Indeed, studies suggested that bats seem to be able to discriminate targets by the variations of the

Doppler effect induced by the wing-flutter rates of the target insects. Such echolocation behavior of microbats can be formulated in such a way that it can be associated with the objective function to be optimized, and this make it possible to formulate new optimization algorithm.

Algorithm:

Bat Algorithm

Objective function $f(x)$, $(x_1, \dots, x_d)^T$

Initialize the bat population x_i ($i = 1, 2, \dots, n$) and v_i

Define pulse frequency f_i at x_i

Initialize pulse rate r_i and the loudness A_i

while ($t < \text{Max number of iterations}$) **do**

generate new solutions by adjusting frequency and updating velocities, locations

if ($\text{rand} > r_i$)

select a solution among the best solutions

generate a local solution around the selected best solution

end if

generate a new solution by flying randomly

if ($\text{rand} < A_i \ \& \ f(x_i) < f(x^*)$)

accept the new solutions

increase r_i and reduce A_i

end if

rank the bats and find the current best x^*

end while

In general the frequency f in a range $[f_{\min}, f_{\max}]$ corresponds to a range of wavelengths $[\lambda_{\min}, \lambda_{\max}]$. For example a frequency range of $[20 \text{ kHz}, 500 \text{ kHz}]$ corresponds to a range of wavelengths from 0.7mm to 17mm. For a given problem, we can also use any wavelength for the ease of implementation.

For simplicity, we can assume $f \in [0, f_{\max}]$. We know that higher frequencies have short wavelengths and travel a shorter distance. For bats, the typical ranges are a few meters. The rate of pulse can simply be in the range of $[0, 1]$ where 0 means no pulses at all, and 1 means the maximum rate of pulse emission [12].

The idealization of the echolocation of microbats can be summarized as follows: Each virtual bat flies randomly with a velocity \mathbf{v}_i at position (solution) \mathbf{x}_i with a varying frequency or wavelength and loudness A_i . As it searches and finds its prey, it changes frequency, loudness and pulse emission rate r . Search is intensified by a local random walk. Selection of the best continues until certain stop criteria are met. This essentially uses a frequency-tuning technique to control the dynamic behavior of a swarm of bats, and the balance between exploration and exploitation can be controlled by tuning algorithm dependent parameters in bat algorithm.

Multi-objective Bat Algorithm (MO-BA):

Using a simple weighted sum with random weights, a very effective but yet simple multi objective bat algorithm (MO-BA) has been developed to solve multi objective tasks. Another multi objective bat algorithm by combining bat algorithm with NSGA-II produces very competitive results with good efficiency.

3. Proposed Work

We can solve the amino acid interaction network prediction problem as well as the protein folding problem using two approaches. The single-objective optimization algorithm (GA) will predict structural motifs of a protein and will give a network or graph of secondary structural element (SSE) of the protein. Then the Artificial Bee Colony (ABC) algorithm will find the interactions between amino acids including the intra-SSE-IN and inter-SSE-IN interactions. In our work we took folded proteins of all- α and all- β class that is already in the Protein Data Bank (PDB) but considered those as unknown sequences and as it has no SCOP family classification.

3.1 Interaction Network Prediction

We can define the problem as prediction of a graph G consist of N number of vertices (V) and E number of edges. If two amino acids interact with each other in protein we mention it as an edge $(u, v) \in E, u \in V, v \in V$ of the graph.

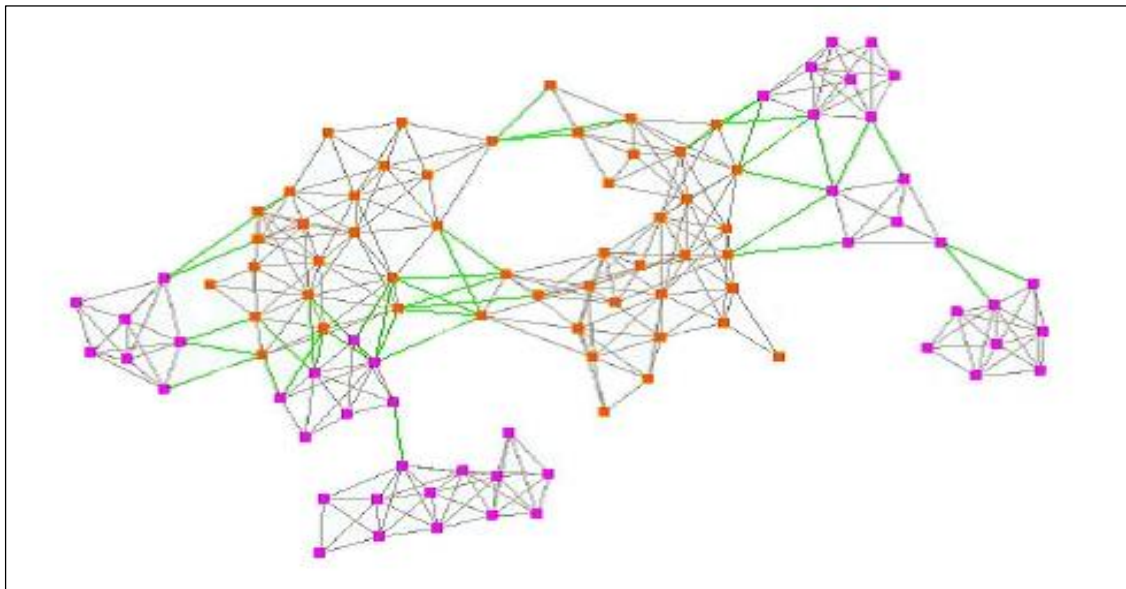


Figure 8: SSE-IN of 1DTP protein. Green edges are to be predicted by artificial bee colony algorithm

A SSE-IN is a highly dense sub-graph G_{SSE-IN} with edge set E_{SSE-IN} . Probability of the edge $(u, v) \in E_{SSE-INA}$, $u \in V_{SSE-INA}$, $v \in V_{SSE-INA}$ is very high and probability of the edge $(u, v) \notin E_{SSE-INA}$, $u \in V_{SSE-INA}$, $v \in V_{SSE-INA}$ is very low where $V_{SSE-INA}$ and $V_{SSE-INB}$ are respectively the vertex set of SSE-IN A and SSE-IN B.

SCOP and CATH are the two databases generally accepted as the two main authorities in the world of fold classification. According to SCOP there are 1393 different folds. To predict the network there are two steps:

- i) Predict a network of amino acid secondary structure element (SSE) from the known SCOP protein family
- ii) Predict interactions between amino acids in the network, including internal edges of SSE-IN and external edges.

3.2 Dataset

We work with the (.pdb) file of different proteins that we obtained from the Protein Data Bank. PDB is a crystallographic database for the three-dimensional structural data of large biological molecules, such as proteins. They are typically obtained by X-ray crystallography or NMR spectroscopy and submitted by biologists and biochemists from around the world and freely accessible on the internet [3].

We took sample proteins from both all- α and all- β class. An all- α proteins is a class of protein domains in which the secondary structure is composed entirely of α -helices, with the possible exception of a few isolated β -sheets on the periphery, and all- β proteins have only β -sheets.

A pdb file contains information about amino acids in atomic level that a protein consists of. It records different atoms with their parent amino acids, 3D coordinate, chain identifier, occupancy etc. We are only interested in C α atoms and it's coordinate. Atom name is in column 13-16 and coordinate is in column 31-54 and it is divided into X (31-38), Y (39-46), Z (47-54) coordinate.

[illegible]

Figure 9: Screenshot of a pdb file with column numbers.

So we need to write a parser that would read a pdb file and extract the Ca atoms and respective coordinate and then measure the distance between two Ca atoms. For parser, we use “*BioJava*”, which is an open-source project dedicated to providing a Java framework for processing biological data. It provides analytical and statistical routines, parsers for common file formats and allows the manipulation of sequences and 3D structures.

3.3 Genetic Algorithm (GA)

Genetic Algorithms (GAs) are adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. As such, they represent an intelligent exploitation of a random search used to solve optimization problems. Although randomized, GAs are by no means random, instead they exploit historical information to direct the search into the region

of better performance within the search space. The basic techniques of the GAs are designed to simulate processes in natural systems.

3.3.1 Overview of GA

In nature, competition among individuals for scanty resources results in the fittest individuals dominating over the weaker ones. GA simulates the survival of the fittest among individuals over consecutive generation for solving a problem. Each generation consists of a population and each individual represents a point in a search space and a possible solution.

Population: For our problem domain, we use GA to predict the amino acids participating in secondary structure, based on a single objective – distance. After extracting C α atoms from a pdb file, we created multiple random array with those atoms. These works as our initial population.

Fitness: after randomly generated our initial population, we calculate fitness for each of them using below equation:

$$\sum_{i=0}^n \text{calculateDistance}(i) \quad (1)$$

Where, n is the length of population array, calculateDistance(i) is a method that takes an index and returns distance between atoms at index (i) and atoms at value of that index. So fitness value of an array is the cumulative sum of distance between each pair of C α atoms.

The individuals in the population are then made to go through a process of evolution with different genetic variation operators.

3.3.2 Genetic Variation Operators

Operator used in genetic algorithms to maintain genetic diversity. Genetic variation is a necessity for the process of evolution. Genetic operators used in genetic algorithms are analogous to those in the natural world.

3.3.2.1 Selection

Selection is the process in which individuals chosen for next step from population of previous step. Our selection procedure is totally random. After calculating fitness values for initial population we select individuals for next step (crossover or mutation) by randomly generated a number and chose individual corresponding to that number.

3.3.2.2 Crossover

Crossover operator is used to create offspring from two parents. The offspring bear the genes of each parent. As a genetic variation operator there is very high probability to crossover occurs other than mutation. In our proposed algorithm we are using uniform crossover.

Parent 1

0	1	1	0	1	0	1	1	1	1
---	---	---	---	---	---	---	---	---	---

Parent 2

1	1	0	1	0	1	1	1	0	0
---	---	---	---	---	---	---	---	---	---

if (Rnd<0.5) copy from parent 1

Offspring

0	1	0	1	1	1	1	1	0	1
---	---	---	---	---	---	---	---	---	---

Figure 10: Uniform Crossover with probability 0.5

Uniform Crossover uses a fixed mixing ratio between two parents. It enables the parent chromosomes to contribute the gene level rather than the segment level. The Uniform Crossover evaluates each bit in the parent strings for exchange with a probability and a probability of 0.5 creates an offspring that has approximately half of the genes from first parent and the other half from second parent.

3.3.2.3 Mutation

Mutation is one of the most widely used variation operator in genetic algorithm. It is used to maintain genetic diversity from one generation of a population to the next and performs the operation in a single individual.

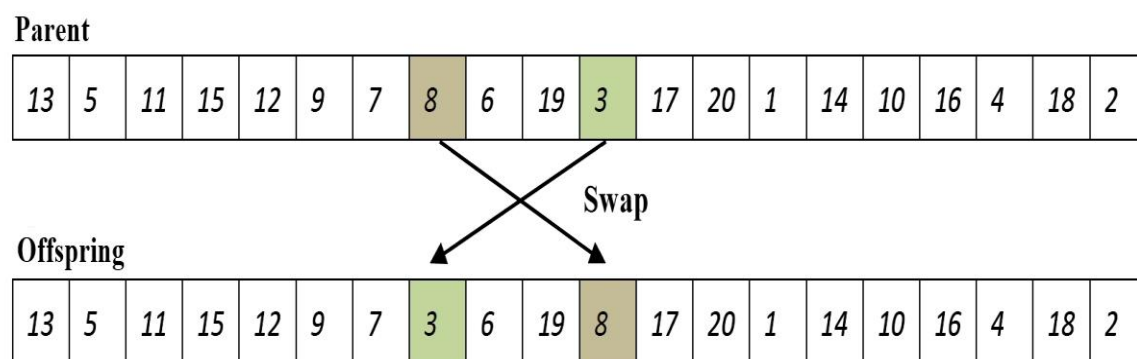


Figure 11: Mutation operator using swap

Mutation alters one or more gene values in a chromosome from its initial state. In mutation, the solution may change entirely from the previous solution. Hence GA can come to better solution by using mutation. Probability of mutation should be set low otherwise the search will turn into a primitive random search. We kept this probability below 0.05.

3.3.3 Pseudocode of GA

```
1:   Input: A pdb file, T = number of steps,  $N_P$  =Population size,  $N_i$  = individual size

2:   Output: individuals with best fitness

3:   generate  $N_P$  random initial population

4:   calculate fitness for all individual using equation (1)

5:   memorize worst fitness,  $W_f$ 

6:   for i = 0 to T do

7:       generate a random number, n in range [0,1]

8:       if n > 0.8 then

9:           for c = 0 to  $N_i$  do

10:                Select two individual randomly

11:                Generate a random probability, p

12:                if p < 0.5 then

13:                    Copy parent 1 data to offspring

14:                else

15:                    Copy parent 2 data to offspring

16:                end if
```

```

17:          end loop

18:      elseif  $n < 0.05$  then

19:          select one individual randomly

20:          generate two random position number, in range  $[0, N_i]$ 

21:          swap value between those two position of individual

22:      else

23:          continue

24:      end if

25:  Calculate offspring fitness,  $f$ 

26:      if  $f < W_f$  then

27:          replace worst individual with offspring

28:      end if

29:  end loop

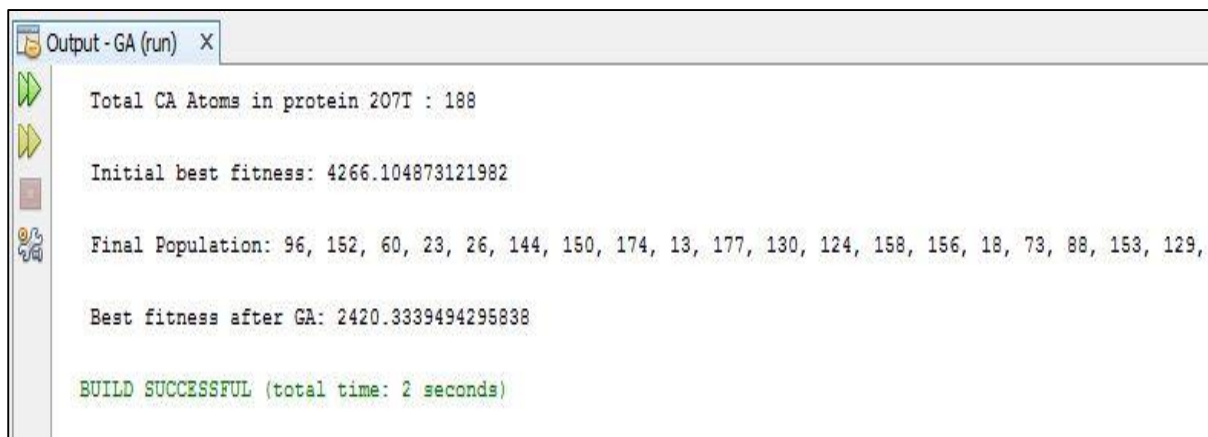
30:  select individual with best fitness

```

3.4 Clustering

After running genetic algorithm for a particular number of iterations, we got our final population with best fitness, which means its cumulative distance is lowest among others. This lowest distance actually means that the amino acid combination represented in the population array, has more probability to be connected.

Figure 12 shows the output that we got after running genetic algorithm. Here we use protein 2O7T pdb file, which contains 188 C α atoms. Initial fitness for randomly generated population was 4266.104Å. After running GA with a step size of 100000 we got our final population with fitness 2420.339Å. This is actually the cumulative sum of all the C α atoms.



```
Output - GA (run) X
>>> Total CA Atoms in protein 2O7T : 188
>>> Initial best fitness: 4266.104873121982
>>> Final Population: 96, 152, 60, 23, 26, 144, 150, 174, 13, 177, 130, 124, 158, 156, 18, 73, 88, 153, 129,
>>> Best fitness after GA: 2420.3339494295838
BUILD SUCCESSFUL (total time: 2 seconds)
```

Figure 12: output of GA using 2O7T protein

Then we apply clustering algorithm on the final population array to get k number of clusters. These clusters will be used as input in our ABC algorithm. Below figure shows a demo clustering with 7 nodes, clustered into 2 clusters.

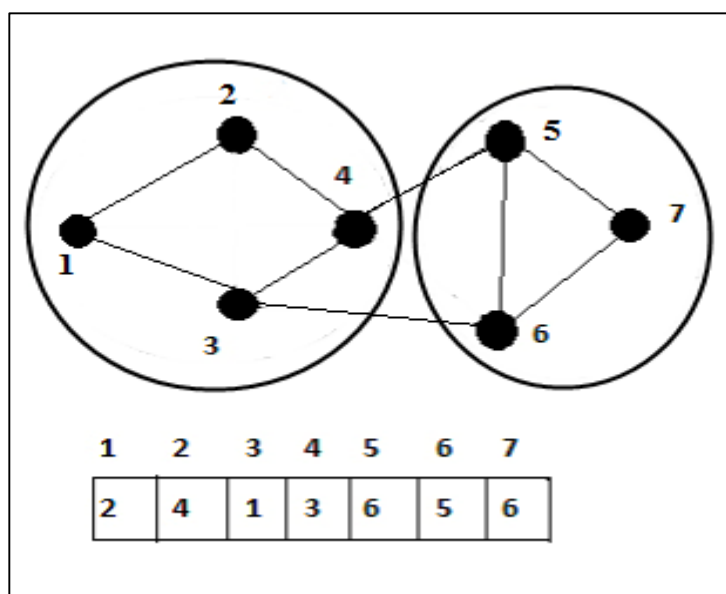


Figure 13: clustering of 7 nodes into 2 clusters

In our approach, we consider each amino acid from GA population array as a node, and try to cluster them. In this representation an individual of the population consist of N amino acids, where N is the number of nodes. It is actually the indexes of the array. Each node can hold value in the range 1,...,N, which are some other amino acids. So both the index of the array and its value represents nodes in the graph $G = (V, E)$ modelling a network N. A value j assigned in i-th node interpreted as a link between node i and j and in clustering node i and j will be in the same cluster, otherwise they form a different cluster.

```

Output - pdb (run) #3 X
run:
Enter the number of clusters:
3

The Final Clusters By Kmeans are as follows:

Group 1[ 53 29 53 14 53 40 21 57 0 57 0 44 58 35 64 17 34 20 3 41 45 14 26 66 53 19 11 53 17 44 53 17 62 57 20 44 44 19 41 41 53 35 30 14 54 19 1

Group 2[ 76 94 75 101 106 75 87 94 122 93 76 116 105 124 125 119 68 122 91 100 94 115 119 76 102 122 92 89 92 93 84 76 102 116 91 82 125 84 102 1

Group 3[ 182 131 165 145 133 182 158 134 171 183 172 149 144 129 144 182 158 182 146 139 146 148 171 164 149 170 172 181 178 179 179 144 166 179

BUILD SUCCESSFUL (total time: 3 seconds)

```

Figure 14: output for clustering algorithm

3.5 Artificial Bee Colony (ABC)

There are so many kind of swarms in the world. It is not possible to call all of them intelligent or their intelligence level could be vary from swarm to swarm. Self-organization is a key feature of a swarm system which results collective behavior by means of local interactions among simple agents. Additional to these characteristics, performing tasks simultaneously by specialized agents, called division of labor, is also an important feature of a swarm intelligence. Social insect colonies can be well thought of as dynamical system collecting information from environment and changing its actions accordingly. While collecting information and adjustment courses, individual insects do not perform all the errands because of their specializations. Usually, all social insect colonies behave according to their own division of labors related to their morphology. Artificial Bee Colony (ABC) is a quite a new member of swarm intelligence. ABC attempts to model natural behavior of honeybees in food searching.

3.5.1 Overview of ABC

Artificial bee colony algorithm is one of the most recently introduced swarm-based algorithm. In ABC system, artificial bees fly around in a multi-dimensional searching space and some (employed and onlooker bees) choose food sources depending on their own experience and that of their nest mates, and adjust their positions.

That means ABC is an optimal tool providing a population-based searching procedure in which individuals called food-position are modified by the artificial bees with time and the bees' targets are to find the food sources with high nectar amount, the place with the highest nectar (the best solution).

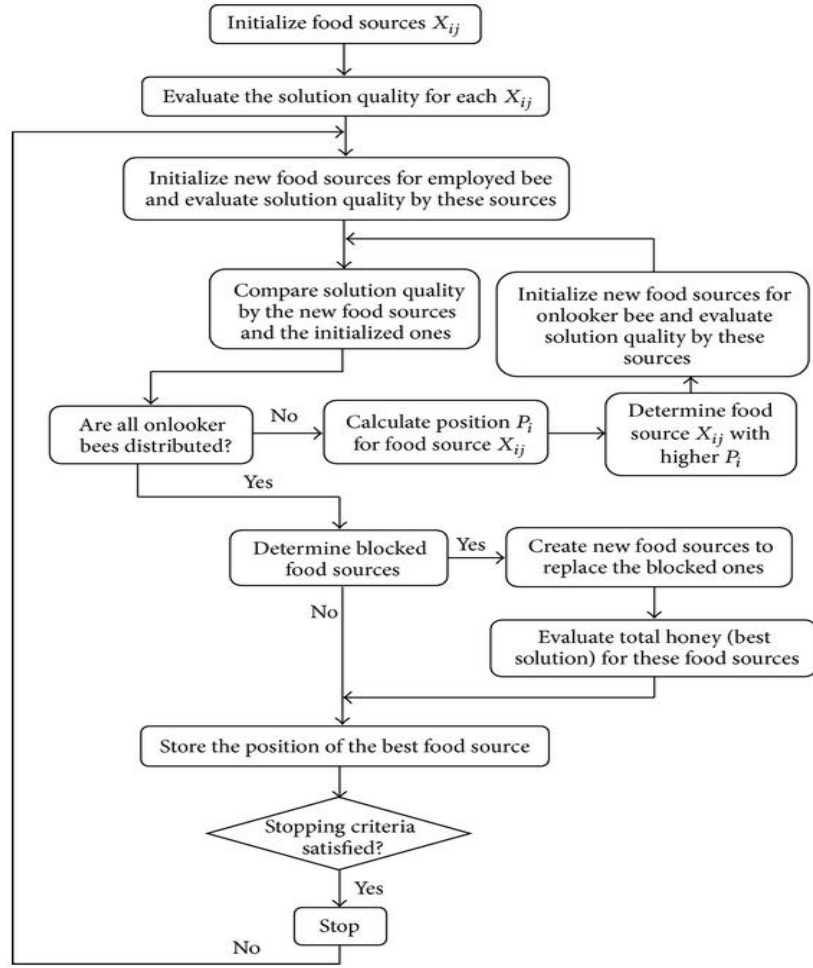


Figure 15: Artificial Bee Colony Algorithm Flowchart

It uses two common control parameters namely colony size and maximum cycle number. ABC system combines local and global search methods attempting to balance exploration and exploitation process.

Bee system consists of three essential components and the model defines two leading modes of the behavior – the recruitment to a rich nectar source and the abandonment of a poor source.

i. Food Sources: The importance of a food source depends on different parameters such as its closeness to the nest, richness of energy and ease of extracting this energy.

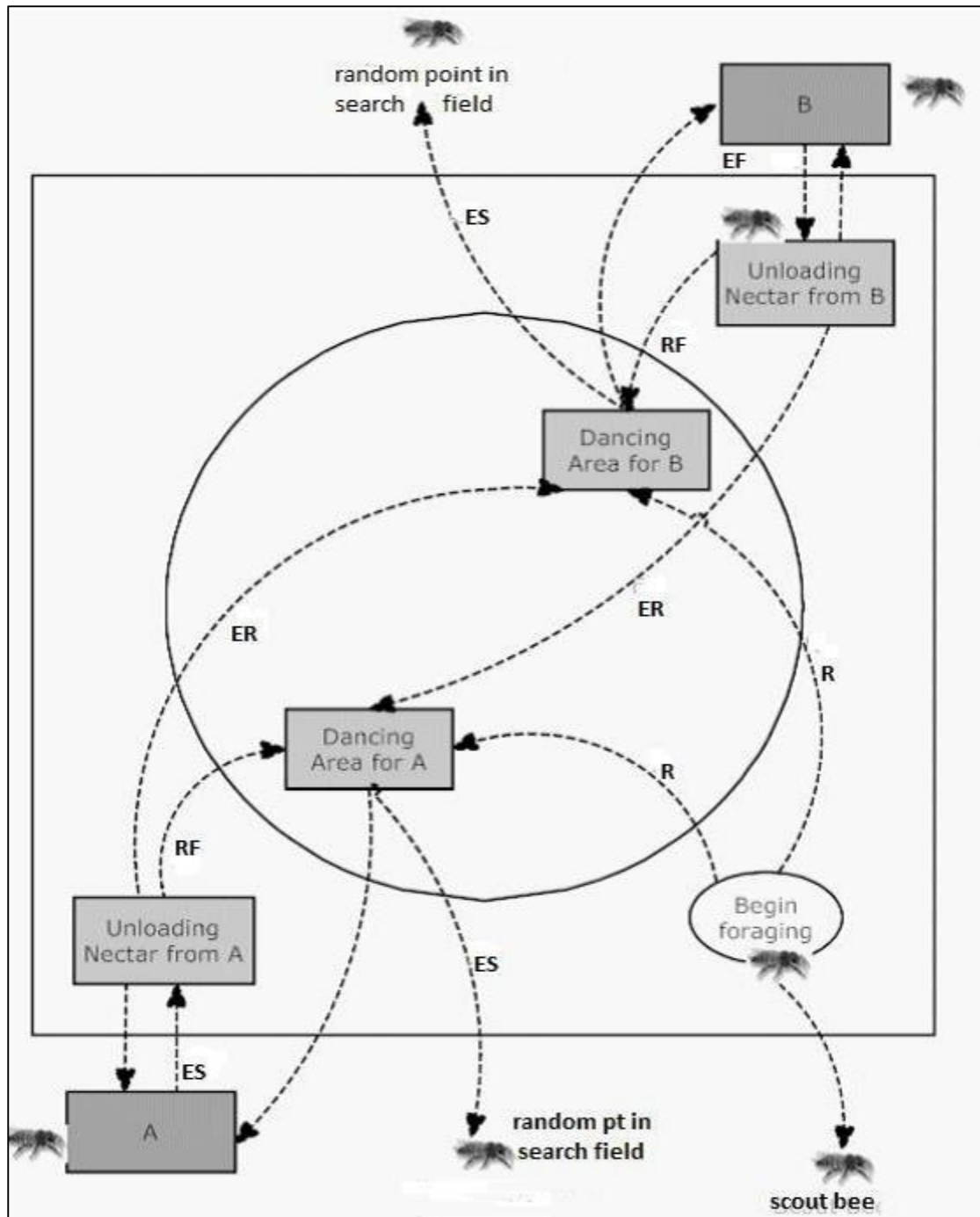


Figure 16: visual representation of ABC

ii. Employed foragers: They are associated with a particular food source which they are currently exploiting or are “employed” at. They carry with them information about this particular source to the hive and the information can be the distance and direction from the

nest, the profitability of the source and share this information with a certain probability. They are showed as EF in figure.

iii. Unemployed foragers: They are continually at look out for a food source to exploit.

There are two types of unemployed foragers: scouts, searching the environment surrounding the nest for new food sources and onlookers waiting in the nest and establishing a food source through the information shared by employed foragers.

3.5.2 How ABC Works

As in the minimal model of forage selection of real honey bees, the colony of artificial bees in ABC contains three groups of bees: *employed bees* associated with specific food sources, *onlooker bees* watching the dance of employed bees within the hive to choose a food source, and *scout bees* searching for food sources randomly.

Initially, all food source positions are discovered by scout bees. Thereafter, the nectar of food sources are exploited by employed bees and onlooker bees, and this continual exploitation will ultimately cause them to become exhausted. Then, the employed bee which was exploiting the exhausted food source becomes a scout bee in search of further food sources once again. In ABC, the position of a food source represents a possible solution to the problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. In the basic form, the number of employed bees is equal to the number of food sources (solutions) since each employed bee is associated with one and only one food source.

In the initialization phase, the population of food sources (solutions) is initialized by artificial scout bees and control parameters are set.

Employed Bees Phase: artificial employed bees search for new food sources having more nectar within the neighborhood of the food source ($x_{i,j}$) in their memory. They find a neighbor food source with below equation:

$$v_{i,j} = x_{i,j} + \phi_{ij} (x_{i,j} - x_{k,j}) \quad (2)$$

Where k = a solution in the neighborhood of i

ϕ = a random number generated using below equation:

$$((double)Math.random() * 32767 / ((double)(32767) + (double)(1)))$$

After producing the new food source, its fitness is calculated with below equation:

$$fitness = \sum_{i=0}^n calculateDistance(i) \quad (3)$$

Then a greedy selection is applied between it and its parent. After that, employed bees share their food source information with onlooker bees waiting in the hive by dancing on the dancing area.

Onlooker Bees Phase: artificial onlooker bees probabilistically choose their food sources depending on the information provided by the employed bees. For this purpose, a fitness based selection technique can be used with below equation:

$$Probability = \frac{fitness_i}{\sum_{i=0}^n fitness_i} \quad (4)$$

After a food source for an onlooker bee is probabilistically chosen, a neighborhood source is determined, and its fitness value is computed. As in the employed bees phase, a greedy selection is applied between two sources.

Scout Bees Phase: employed bees whose solutions cannot be improved through a predetermined number of trials, called “limit”, become scouts and their solutions are abandoned. The food source of which the nectar is abandoned by the bees is replaced with a new food source by the scouts. In ABC, this is simulated by producing a position randomly and replacing it with the abandoned one. The value of predetermined number of cycles is an important control parameter of the ABC algorithm, which is called “*limit*” for abandonment. Assume that the abandoned source is x_i and $j \in \{1, 2, 3, \dots, D\}$, where D is the number of optimization parameters. Then the scout discovers a new food source to be replaced with x_i using below equation:

$$x_i^j = x_{min}^j + rand(0,1) (x_{max}^j - x_{min}^j) \quad (5)$$

After each candidate source position v_j^i is produced and then evaluated by the artificial bee, its performance is compared with that of its old one. If the new food has an equal or better nectar than the old source, it is replaced with the old one in the memory. Otherwise, the old one is retained in the memory.

These three steps are repeated until a termination criteria is satisfied, for example a maximum number of steps.

3.5.3 A Proposed Crossover Operator

The GA has a very important operator that ABC doesn't have, which is the crossover operator. The cooperative ability of GA approach gave it the advantage of modeling cooperative pathways. Thus, it is effective to use GA features in protein folding problem because the protein folding is not cooperative on the dynamic level only but it is cooperative on the interaction level as well.

According to [13], the crossover operation is the heart of the method. By crossover a new region between the two parents is available for search, as it was not accessible previously by two parent solutions. Cooperation between solutions has proved to have a very positive effect on the efficiency of the search algorithm.

For this, a crossover operator was added as an additional phase to the three main phases of the bee colony algorithm. The crossover phase has been added after the employed bee phase to complement the differential search convergence. We used simulated binary crossover here with below equation:

$$C(\beta) = 0.5 (n + 1)\beta^n \quad (6)$$

Moreover, a greedy selection was used in the crossover operator. After selecting the parents according to their fitness, uniform crossover is applied and a pool of new offspring is generated. Because of the greedy selection process, these new offspring competes with its parents for the next generation and the one with the better fitness wins.

For the PSP problem, the crossover is represented by using an array containing distance of all initial population. This is represented as D dimensions for food source in ABC algorithm.

3.5.4 Pseudocode of ABC

- 1: **Input:** population size SN, number of steps MCN, number of optimization parameters D, limit, clustered network
- 2: **Output:** predicted edges of network
- 3: Initialize the population of solutions $x_{i,j}$, $i = 1 \dots SN$, $j = 1 \dots D$
- 4: Evaluate the population using equation (3)
- 5: **for** cycle=1 to MCN **do**

 //employed bee phase
- 6: produce new solutions $v_{i,j}$ for the employed bees by using equation (2)
- 7: evaluate new solutions
- 8: **if** $f(v_{i,j}) < f(x_{i,j})$ **then**
- 9: $x_{i,j} = v_{i,j}$
- 10: $f(x_{i,j}) = f(v_{i,j})$
- 11: set trial to 0
- 12: **else**
- 13: increase trial by 1
- 14: **end if**
- 15: apply simulated binary crossover on new solutions $v_{i,j}$ using equation (6)

 //onlooker bee phase
- 16: calculate the probability values $P_{i,j}$ for the solutions $x_{i,j}$ by using equation (4)

17: select the new solutions $\mathbf{v}_{i,j}$ for the onlookers from $\mathbf{x}_{i,j}$, depending on $P_{i,j}$

18: evaluate new solutions

19: **if** $f(\mathbf{v}_{i,j}) < f(\mathbf{x}_{i,j})$ **then**

20: If $\mathbf{x}_{i,j} = \mathbf{v}_{i,j}$

21: If $f(\mathbf{x}_{i,j}) = f(\mathbf{v}_{i,j})$

22: set trial to 0

23: **else**

24: increase trial by 1

25: **end if**

 //scout bee phase

26: **if** trial > limit **then**

27: generate a new randomly produced solution $\mathbf{x}_{i,j}$ by using equation (5)

28: replace abandoned solution with new one

29: **end if**

30: Memorize the best solution achieved so far

31: **end loop**

4. Performance Analysis

4.1 Analysis of Genetic Algorithm

In order to test the performance of genetic algorithm, we randomly pick four individual from the final population and we compare their associated matrices with the SSE-IN adjacency matrix. To evaluate the difference between two matrices, we use an error rate defined as the number of wrong elements divided by the size of the matrix. All alpha class has an average error rate of 19.7% and for the all beta class it is 16.2% and the maximum error rate shown in the experiment is 24.6%. The error rate depends on other objectives like hydrophilic/hydrophobic property of amino acids, torsion angle between two amino acids etc. but according to the result we can say that the error rate depends on the number of initial population and iteration steps. More the number of initial population and step size, less the error rate. With sufficient number of individuals in the initial population we can ensure the genetic diversity as well as the improved SSE-IN prediction. When the number of initial population is at least 15 and step size 50,000, the error rate is always less than 15%.

4.2 Analysis of ABC Algorithm

We have tested this part of our proposed method according to the associated family protein because the probability of adding edge is determined by the family occurrence matrix. This information can be obtained from SCOP website, which is protein classification website and we call this associated family protein as template protein.

For each protein, we have done around 150 simulations and when the topological properties are become compatible to the template properties of the protein we accepted the built SSE-IN. after each simulation, we took the resultant array that is generated by ABC algorithm, make a graph from it and compute the diameter, path length, mean degree etc. graph properties and match those with the properties of template ones. Here we allow an error up to 20% to accept the built sequence SSE-IN. If the built SSE-IN is not compatible, it is rejected and next simulation is done.

For measuring the accuracy of graph properties, we compare the predicted value (E_p), with the template ones value (E_T) –

$$AC = 1 - \frac{|E_T - E_p|}{E_p}$$

Global interaction prediction rate depends on the local algorithm done for each pair of SSEs in contact. If the local algorithm select at least 80% of the correct edges, then the global interaction rate stays better than the 80% and evolve around 85% for the all alpha class and 73% for the all beta class. The average score decreases for big proteins (of size more than 200). For these proteins there are many pairs of SSE in contact and the errors made by the local algorithm accumulate to global level.

4.3 Algorithm Complexity

Two algorithms we proposed are independent of specific time bound. Both the optimization algorithm used as genetic algorithm and artificial bee colony algorithm, is iteration based. We

can stop the algorithm at any time as we define the step size. The result of the algorithm depends on the number of iteration but if we give sufficient amount of iteration it provides good result. In compare to other state of the art algorithms, that uses exponential complexity algorithm, ours one is linear in terms of time and memory.

4.4 Result

Table:

<i>Class</i>	<i>SCOP Family</i>	<i>Number of Proteins</i>	<i>Protein Size</i>	<i>Score (ABC)</i>	<i>Score (SBX ABC)</i>	<i>Score (DE ABC)</i>
All Alpha	46688	17	27-46	81.656	84.321	78.032
	47472	10	98-125	75.482	77.876	74.832
	46457	25	129-135	71.921	74.762	67.491
	48112	18	241-281	68.345	70.882	66.110
	48507	20	387-422	62.579	69.479	60.769
All Beta	50629	6	54-66	78.503	81.552	78.241
	50813	11	90-111	76.332	78.956	73.728
	48725	24	120-124	75.874	77.115	71.591
	50629	13	124-154	72.115	74.890	69.882
	50875	14	155-224	69.984	73.665	67.423

Graph:

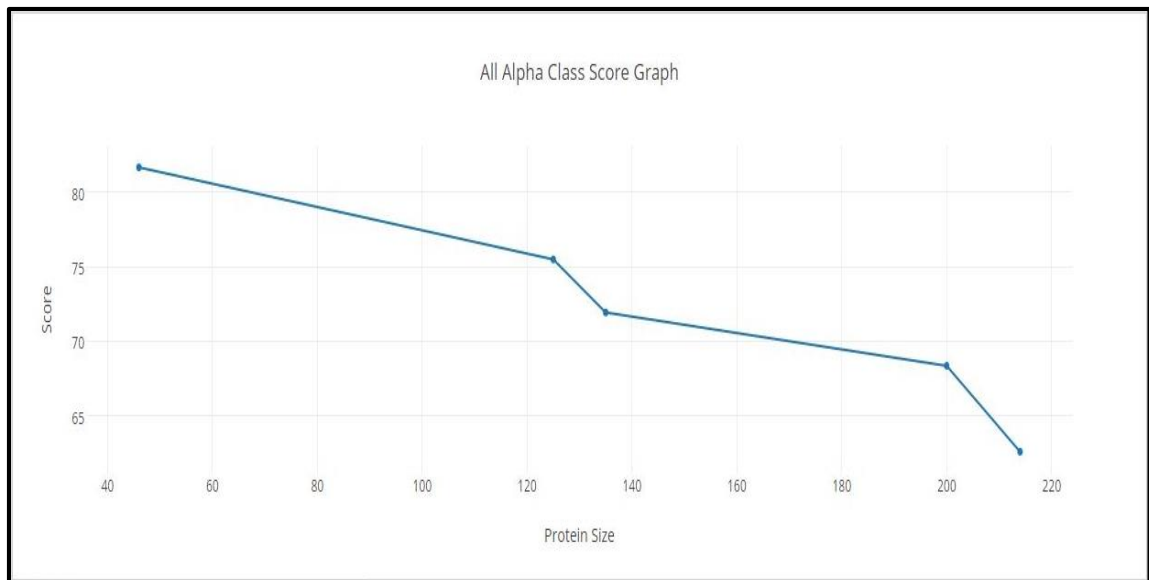


Figure 17: All alpha class score decreasing with protein size

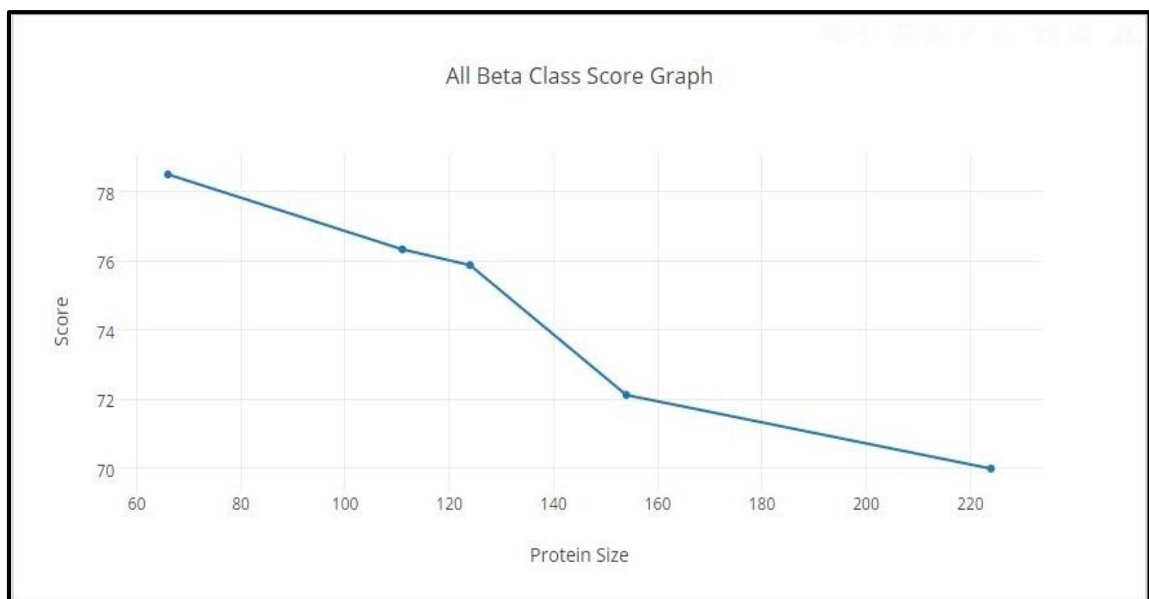


Figure 18: All beta class score decreasing with protein size

Score Graph using ABC with SBX:

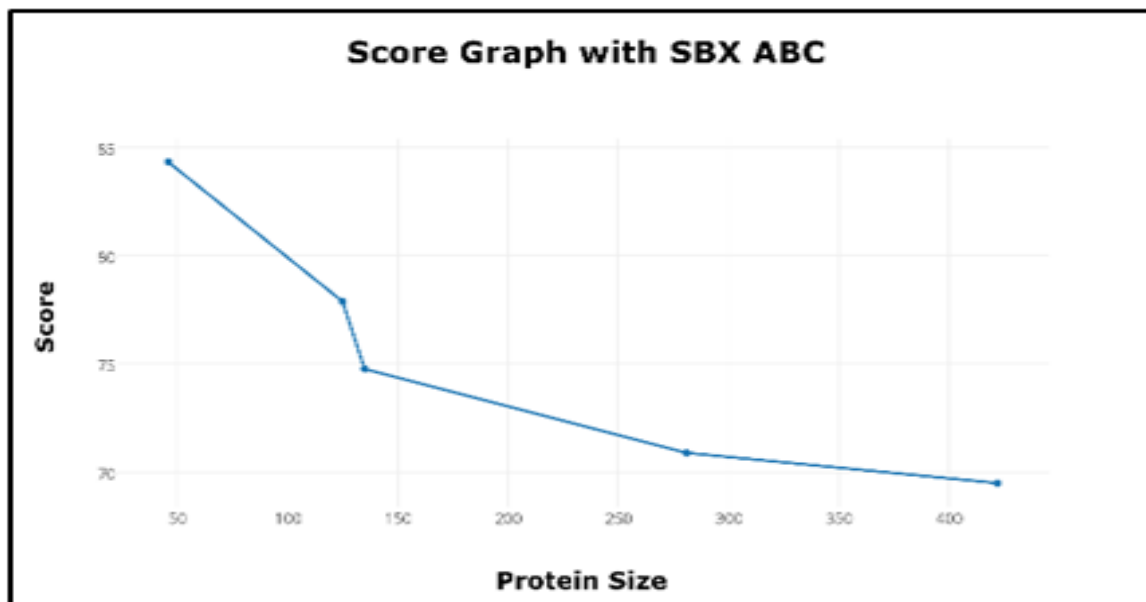


Figure 19: All alpha class score decreasing with protein size (SBX ABC)

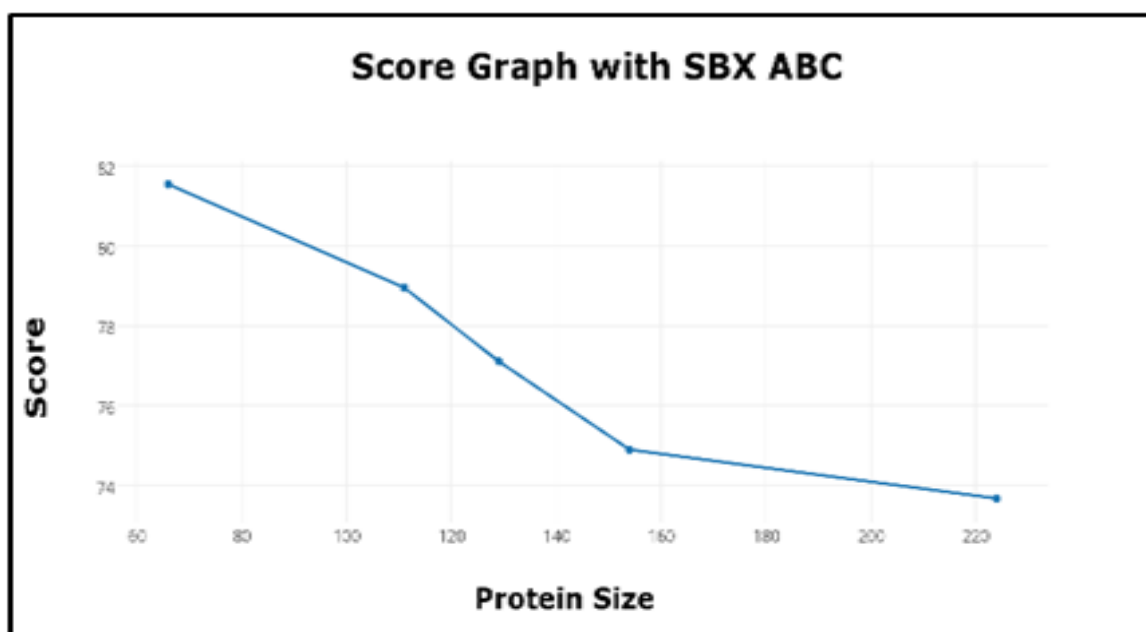


Figure 20: All beta class score decreasing with protein size (SBX ABC)

Score Graph using ABC with DE:

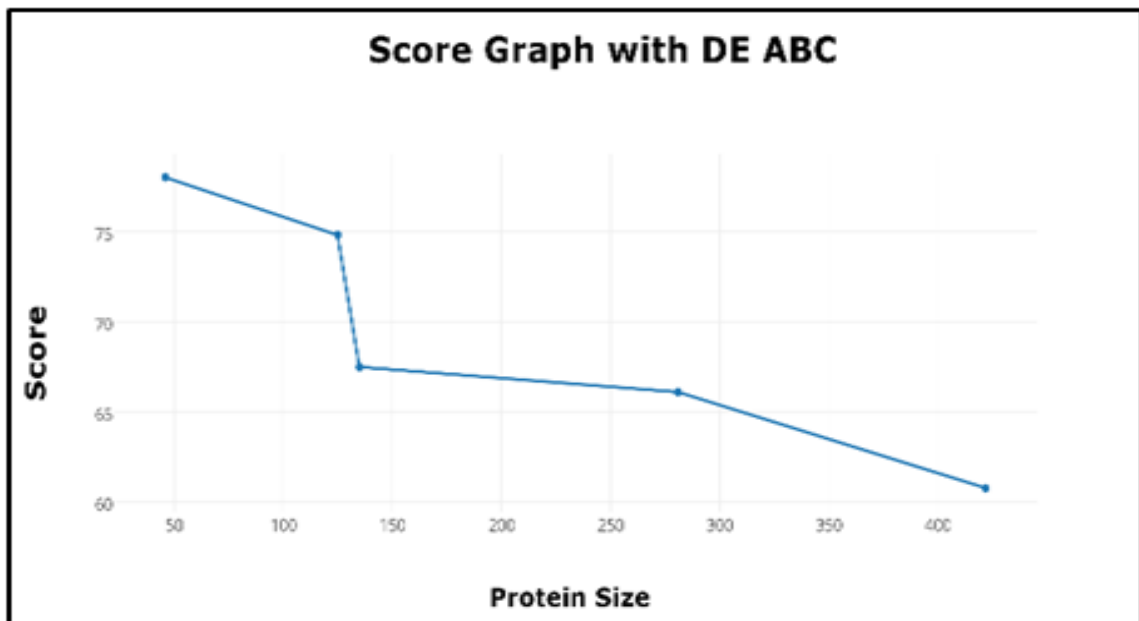


Figure 21: All alpha class score decreasing with protein size (DE ABC)

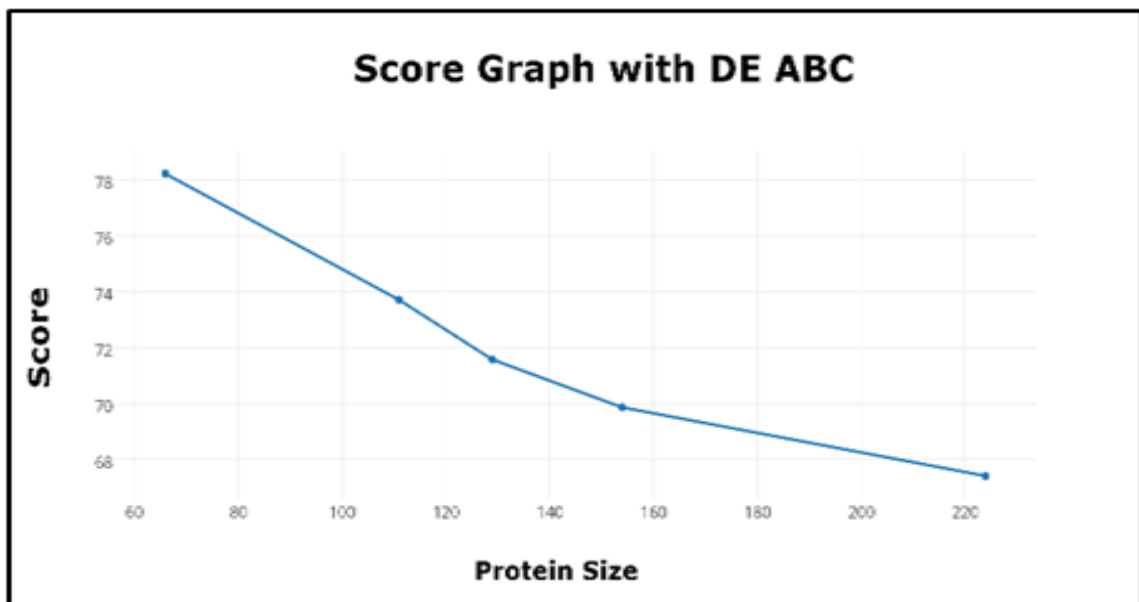


Figure 22: All beta class score decreasing with protein size (DE ABC)

5. Future Work

In our proposed work, we used genetic algorithm with single objective as the distance between to amino acid in protein atom. But it is very difficult to define real world problems like amino acid interaction prediction problem in terms of a single objective. A multi-objective optimization problem deals with more than one objective functions that are to be minimized or maximized. These objectives can be conflicting, subject to certain constraints and often lead to choosing the best trade-off among them. The interaction between amino acids in protein depends not only distance between two amino acids but also the torsion angles and hydrophobic property of the amino acid.

The two torsion angles of the polypeptide chain, also called Ramachandran angles, describe the rotations of the polypeptide backbone around the bonds between N – C α (called Phi angle, ϕ) and C α – C (called Psi angle, Ψ). These torsion angles are one of the most important parameter of protein structure and controls the protein folding. Another important property of protein is hydrophobicity. Proteins tertiary structure's core are hydrophobic and the amino acids inside core part do not interact much as like their counterpart hydrophilic, those made the outer side of the protein structure.

So to get more accurate interaction network of amino acid we have to consider it is as a multi objective problem rather than single objective. For our future work, we plan to use –

- multi objective genetic algorithm
- improved clustering algorithm
- newly emerged Bat algorithm for interaction prediction

6. Conclusion

We have proposed a computational solution to a biological problem. In our research we have described how we can formulate protein folding problem into optimization and graph theory problem. The formulation consists of finding the interactions between secondary structure element (SSE) network and interaction between amino acids of the protein. We addressed the first problem using genetic algorithm and the second one solved by artificial bee colony optimization algorithm. Results shows that our proposed approach has around 80% accuracy, though it can be furnished further with improved approaches as we stated in our future plan.

7. References

1. E. Pennisi, 'HUMAN GENOME: A Low Number Wins the GeneSweep Pool', *Science*, vol. 300, no. 5625, pp. 1484b-1484, 2003.
2. D. Karaboga, "An idea based on honey bee swarm for numerical optimization" TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.
3. Murzin AG, Brenner SE, Hubbard TJP, Chothia C. 1995. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology* 247:536-540.
4. H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, P.E. Bourne (2000) The Protein Data Bank, *Nucleic Acids Research*, 28: 235-242.
5. Rcsb.org, 2015. [Online]. Available: <http://www.rcsb.org/pdb/files/2O7T.pdb>. [Accessed: 07- Mar- 2015].
6. A. Ghosh, K.V Brinda, S. Vishveshwara. "Dynamics of lysozyme structure network: probing the process of unfolding". *Biophysical journal*, 92(7):2523–2535, 2007.
7. O. Gaci, S. Balev. "The small-world model for amino acid interaction networks", In *Advanced Information Networking and Applications Workshops*, 2009.

8. Gaci, Omar. “Building a topological inference exploiting qualitative criteria”. Evolutionary Bioinformatics, 2010.
9. Dorigo, M., Maniezzo, V. & Coloni, A. (1996), “Ant system: Optimization by a colony of cooperating agents”, IEEE Transactions on Systems, Man, and Cybernetics – Part B, Cybernetics 26, 29–41.
10. Holland, J.H., “Adaptation in Natural and Artificial Systems”, University of Michigan Press, Ann Arbor, 1975.
11. M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, 2004.
12. X.-S. Yang, “A New Metaheuristic Bat-Inspired Algorithm”, in: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010) (Eds. J. R. Gonzalez et al.), SCI 284, 65-74 (2010).
13. Unger, R. “The genetic algorithm approach to protein structure prediction”, In Applications of Evolutionary Computation in Chemistry, pp. 153-175 (2004).
14. M. Levitt and C. Chothia. “Structural patterns in globular proteins”. Nature. 261:552–558, 1976.
15. C. Branden and J. Tooze. “Introduction to protein structure”. Garland Publishing, 1999.